Segment Trees based Traffic Congestion Avoidance in Connected Cars Context

Ioan Stan*, Dan Toderici†, Rodica Potolea‡
Technical University of Cluj-Napoca
Computer Science Department
Romania
*E-mail: *ioan.stan@cs.utcluj.ro,*
†todericidan@mail.student.utcluj.ro,
‡rodica.potolea@cs.utcluj.ro

Abstract—**Nowadays traffic congestion in cities is one of the main challenges that drivers are facing. Cities administration doesn't always manage to handle this challenge with success and the increasing number of cars makes the situation worse. Navigation Systems, besides generating routes between a starting and ending point, can be used to predict and avoid traffic congestion.**

In this paper we propose a novel solution for traffic information representation in connected cars context. The strategy is based on segment trees data structure and was integrated into an industry navigation system by enhancing routing algorithm to support traffic congestion prediction and avoidance.

The experimental results proves that connected cars information can be used to predict and optimize traffic flow in a city.

1. Connected Cars Navigation Systems

Internet of Things (IoT) concept encapsulates and combine large technological domains that can affect human life in a positive or negative way. Such a domain is Intelligent Transportation System that can be seen as a combination of infrastructure, computing, telecommunications, wireless and transportation technologies [1]. A review of the state of the art in the Intelligent Transportation System development and vehicular communication is done in [2] and [3]. In [4] Miller describes and analyses a hierarchical architecture of the Intelligent Transportation System also known as Vehicle-to-Vehicle-to-Infrastructure (V2V2I).

Connected Cars and especially their Navigation System are main parts of an Intelligent Transportation System. Vehicular communication and data sharing can improve services that are based on Navigation System (emergency services - see [1]). Moreover, as is shown in this paper, efficient data sharing and representation in the context of Connected Cars Navigation System results in traffic congestion avoidance and therefore, reduction of time spent in traffic. Besides traffic information, one of the main components of a Navigation System is the Routing Algorithm that can be directly used to support traffic congestion avoidance.

2. Routing Algorithms in Navigation Systems

Route computation is generally based on graph algorithms and variations of them (.e.g. Dijkstra, A*). As time passed and navigation systems becomes more complex and the routing algorithms required enhancements in order to fulfill expectations. Many routing algorithms challenges (e.g. computation time, memory usage, route alternatives) were solved successfully. In [5] are analyzed and compared different routing algorithms and showed that there exists very fast routing algorithms (e.g. 1Mx faster than Dijkstra) that can be used in the industry. At least at the scientific level, fast route computation algorithms is not anymore a challenge.

The reach-based routing algorithm presented in [6] have a simpler implementation alternative that is combined in an elegant way with A* graph search [7].

Most of the routing algorithms are using highway hierarchies to improve their computation time. In [8] is described and analyzed the impact of using highway hierarchies for many-to-many shortest route calculation.

Besides route computation time, a more important actual issue is the resulted route driving time and the estimated time of arrival (ETA) for such a route. Reducing driving time for the cars that run on the roads will be a high benefit for each driver. Existing solutions in the industry are based on classic graph algorithms (e.g. Dijkstra, A*) and computes ETA using the geometry of the road and traffic events as parameters. Moreover, the routing algorithms try to solve the problem of driving time by generating the fastest route for a specific request considering the route geometry and traffic information and doesn't take into account the entire ecosystem state.

3. Routing Algorithms for Traffic Prediction and Avoidance

Traffic congestion is one of the challenges that drivers encounter day by day. As number of cars are increasing, the roads in the cities become crowded and citizens lose their time in traffic. One direct solution to this problem is to reduce the number of cars that passes cities and enlarge the existing roads. This solution have many dependencies and unknowns (drivers, local administration, public transportation, infrastructure, etc.) that must be solved and this can take too much time for a generation.

Most of the existing navigation systems that consumes map data and traffic information during route computation can be enhanced to be able to generate routes systematically in a way that all roads usage is balanced and traffic congestion on principal roads is avoided as much as possible. To the best of our knowledge we suppose that only Waze navigation system uses all these concepts during route generation: real map data, traffic information, "connected cars information data" that is represented by the position of the mobile devices on which the navigation system is running. This is not confirmed in any document but is a supposition of the authors based on the navigation system behavior and on the information found in [9].

Our approach is an enhancement of an open source existing industry navigation solution OSMAnd (see [10]) that generates routes based on real map data. OSMAnd is based only on map data without considering traffic information during route calculation.

On top of the real map data, our routing strategy relies on connected cars data, where the traffic information is simulated in a systematic way that tries to mimic real world traffic. Relying on map data and connected cars information our strategy aims to predict and avoid traffic congestion at a global level and not only at an individual level.

3.1. Basic Routing Algorithm

Usually, navigation systems that are based on real map data and traffic information have a routing algorithm that uses a set of parameters as described below. In our solution, these parameters are integrated in a bidirectional (forward and backward) search flow that follows A* strategy [11], [12]. Further in this paper we will name such algorithm Basic Routing Algorithm (BRA).

BRA main parameters and helper functions are described below.

- $\mathbf{p(x, y)}$ - GPS point represented by x and y coordinates
- $\mathbf{p_s}$ - starting point of a route
- $\mathbf{p_d}$ - destination point of a route
- $\mathbf{turn_i}$ - turn costs that have a fixed value for a specific turn type (e.g. right turn, slight right turn, left turn, etc.)
- $\mathbf{segID}$ - segment represented by segment ID
- $\mathbf{S_{limit}(segID))}$ - default speed limit of a segment
- $\mathbf{C_{map}(segID)}$ - map cost value of a segment ID based on map geometry (e.g. euclidian distance, turn costs, speed limit on the segment, etc.)
- $\mathbf{R(p_s, p_d)}$ - fastest generated route between starting point p_s and destination point p_d
- $\mathbf{N(segID)}$ - set of neighbour segments of segID
- $\mathbf{visited(segID)}$ - returns TRUE if segID was visited in the routing algorithm search graph and FALSE otherwise
- $\mathbf{T(segID, t)}$ - predicted traffic on a segment at a specific time
- $\mathbf{C_{traffic}(traffic(segID, t))}$ - cost factor corresponding to predicted traffic on a segment at a specific time

Algorithm 1 shows the main flow of the Basic Routing Algorithm (BRA) for fastest route. It is a bidirectional A* algorithm that predicts traffic on a segment at a specific time and uses this information during route calculation.

Algorithm 1 Basic Routing Algorithm (BRA)

```
Data: p_s, p_d, map_data, traffic_service
Result: R(p_s, p_d)
forwardQueue ← init forward search graph queue
backwardQueue ← init backward search graph queue

if forwardSearch then
  | queue ← forwardQueue
else
  | queue ← backwardQueue
end
while forward and backward search unmet do
  currentSegID ← queue.head()
  foreach segID in $N(currentSegID)$ do
    if [not $visited(segID)$ then
      $C_{map} \leftarrow C_{map}(segID)$
      t ← predicted time when the car arrive on segment
      $C_{traffic} \leftarrow C_{traffic}(T(segID, t))$
      $costValue \leftarrow C_{map} \cdot C_{traffic}$
      $updateCost(segID, costValue)$
    end
  end
end
```

3.2. Connected Cars Routing Algorithm

In the context of connected cars, besides the above mentioned parameters of BRA in order to generate a route that avoids future traffic congestion a set of new parameters have to be considered:

- $\mathbf{carID}$ - car represented by an ID. It corresponds to a route $R(p_s, p_d)$
- $\mathbf{R(carID)}$ - route on which a car is supposed to follow
- $\mathbf{Routes(t)}$ - routes set generated until a specific time
- $\mathbf{Segments(R(carID)}$ - set of segments of a route
- $\mathbf{lanesCount(segID)}$ - number of lanes on a segment used for car density computation

- $[\mathbf{t_s}(\mathbf{carID}, \mathbf{segID}), \mathbf{t_e}(\mathbf{carID}, \mathbf{segID})]$ - predicted time interval when a car is on a segment represented by segment ID
- $\mathbf{carsCount}(\mathbf{segID}, \mathbf{t})$ - predicted number of cars on a segment at a specific time used for car density computation
- $\rho(\mathbf{segID}, \mathbf{t})$ - predicted car density on a segment at a specific time
- $\mathbf{S_{avg}}(\mathbf{segID}, \mathbf{t})$ - average speed on segments at a specific time
- $\mathbf{C_{cars}}(\rho(\mathbf{segID}, \mathbf{t}))$ - cost factor corresponding to predicted car density on a segment at a specific time
- θ - threshold value that indicates traffic congestion

Algorithm 2 shows the main flow of the Connected Cars Routing Algorithm (CCRA) for fastest route. The CCRA enhancement over BRA is the decision based on connected cars future traffic congestion replacing the current traffic penalty used during cost segment calculation in BRA. In this regards, CCRA tries to avoid traffic congestion by forcing an alternative route generation that balances segments usage in the map when the car density $\rho(segID, t)$ reaches a threshold θ.

Algorithm 2 Connected Cars Routing Algorithm (CCRA)

```
Data: p_s, $p_d, map\_data, connected\_car\_data, \theta$
Result: R(p_s, $p_d$)
forwardQueue ← init forward search graph queue
backwardQueue ← init backward search graph queue

if forwardSearch then
    queue ← forwardQueue
else
    queue ← backwardQueue
end
while forward and backward search unmet do
    currentSegID ← queue.head()
    foreach segID in $N(currentSegID)$ do
        if [not $visited(segID)$ then
            $C_{map} \leftarrow C_{map}(segID)$
            $t \leftarrow$ predicted time when the car arrive on segment
            if $\theta < \rho(segID, t)$ then
                $C_{cars} \leftarrow \infty$  ▷ force to try another segment
            end
            else
                $C_{cars} \leftarrow C_{cars}(\rho(segID, t))$
            end
            $costValue \leftarrow C_{map} \cdot C_{cars}$
            $updateCost(segID, costValue)$
        end
    end
end
```

In both, BRA and CCRA, the traffic prediction for forward search has a very good precision because it is easy to predict (by computation) the time spent from p_s until a point on a route $R(p_s, p_d)$. For backward search, the traffic prediction precision is not very good because the arrival time on a segment in backward search graph is roughly estimated based aerial distance and S_{limit} on segments.

In the BRA and CCRA we used just some of the mentioned parameters in order to simplify the flow presentation but we let the parameters description in order to have a clear view of the context for both algorithms.

4. Segment Trees

Predicting and using traffic information in a connected cars context requires to efficiently answer questions as below

- *"What is the traffic on a segment from a route in the time interval [t1, t2] ?"*
- *"What are the number of cars on a segment from a route in the time interval [t1, t2] ?"*

We claim that segment trees is the right data structure to answer the questions above (see [13] for more details about segment trees.). For our purpose, queries can be answered in $O(\log_n)$, where n is the number of time units for which we want to predict traffic/number of cars. To the best of our knowledge this is a novel solution used to represent traffic information in a connected cars context.

The $O(\log_n)$ Algorithm 3 shows update procedure for segment trees used to predict traffic information and number of cars. One $node$ of the interval tree, represented by $[left_{node}, right_{node}]$ interval, increases its value if it overlaps with the update interval request represented by $[left_{interval}, right_{interval}]$.

A key observation for this algorithm is the fact that we have to count if an update interval overlaps both children intervals of a node. As is shown in Algorithm 4 this value is decreased when a queried interval overlaps both node's children. In this way we avoid to double count traffic information that overlaps their active time on a segment with both children intervals of a parent node.

The segment tree query procedure shown in Algorithm 4 return the traffic information in a connected cars context that corresponds to a segment for a specific query interval. The input parameters are the same as for Algorithm 3 and the returned result represent the connected cars traffic information on a specific segment during a time interval. As was explained above it pays attention to not double count information that overlaps both children intervals of a parent node.

Navigation systems in general uses traffic services to get traffic information to be used during route generation process. In our approach, besides the efficiency of segment trees data structure in terms of storing and querying connected cars traffic information, we found that segment trees offer proper support to adapt the initial traffic agnostic routing algorithm to a routing strategy that uses connected cars traffic information.

Algorithm 3 updateSegmentTree

```
Data: node, left_{node}, $right_{node}, left_{interval}, right_{interval}$
if $left_{interval} > right_{node}$ or $right_{interval} < left_{node}$ then
    return        ▷ node doesn't overlap with update interval
▷ update node data that represents traffic information
  in connected cars context
data[node] ← data[node] + 1
if $left_{node} < right_{node}$ then
    $middle \leftarrow \frac{left_{node}+right_{node}}{2}$
    $updateSegmentTree(2 \cdot node, left_{node}, middle,$
    $left_{interval}, right_{interval})$
    $updateSegmentTree(2 \cdot node + 1, middle + 1,$
    $right_{node}, left_{interval}, right_{interval})$
    if $left_{inteval} \leq middle$ and $right_{interval} > middle$ then
        ▷ update interval overlaps both node's children
        $commonData[node] \leftarrow commonData[node] + 1$
    end
end
```

Algorithm 4 querySegmentTree

```
Data: node, left_{node}, $right_{node}, left_{interval}, right_{interval}$
Result: connected cars traffic information on a segment in
        during a time interval [left_{interval}, $right_{interval}$]
if $left_{interval} \leq left_{node}$ and $right_{interval} \geq left_{node}$ then
    return data[node]     ▷ node included in query interval
if $left_{interval} > right_{node}$ or $right_{interval} < left_{node}$ then
    return 0;     ▷ node doesn't overlap with query interval
$middle \leftarrow \frac{left_{node}+right_{node}}{2}$
$leftData \leftarrow querySegmentTree(2 \cdot node, left_{node},$
$middle, left_{interval}, right_{interval})$
$rightData \leftarrow querySegmentTree(2 \cdot node + 1,$
$middle + 1, right_{node}, left_{interval}, right_{interval})$
$value \leftarrow leftData + rightData$
if $left_{inteval} \leq middle$ and $right_{interval} > middle$ then
    $value \leftarrow value - commonData[node]$
end
return value;
```

5. City Traffic Measurements and Experiments

Our solution's main goals is to improve the traffic flow in crowded cities. In this regards, we run, analyzed and compared BRA vs. CCRA on a scenario that simulates traffic in a city.

5.1. Measurements Infrastructure

The base application for our solution is OSMAnd. It is an industry open source navigation solution implemented in Java that can run on Android, iOS and desktop platforms [10].

OSMAnd routing algorithm is based on bidirectional A* and only uses map data information during route calculation. The routing algorithm is not designed to use traffic information during route calculation.

To achieve the purpose of this work we changed the routing algorithm implementation to take into account connectedf cars traffic informatio during route generation process.

5.2. Metrics

In our scenario the route request generation represents fastest mode. Below are described the metrics we used for BRA vs. CCRA comparison.

- $\mathbf{ETA(carID)}$ - Estimated time of arrival for each route request represented by a carID. This is equivalent with computed route driving time.
- $\mathbf{ETA_{BRA}})$ - Sum of ETAs for BRA. Using this metric we want to see what is the total driving time in a city by using BRA approach
- $\mathbf{ETA_{CCRA}})$ - Sum of ETAs for CCRA. Using this metric we want to see what is the total driving time in a city by using CCRA approach
- $\mathbf{speed(carID)}$ - average speed of a specific car corresponding to a generated route (BRA vs. CCRA)
- $\mathbf{Speed_{BRA}}$ - car average speed for BRA
- $\mathbf{Speed_{CCRA}}$ - car average speed for CCRA
- $\mathbf{Segments_{BRA}}$ - total number of touched segments in the map for BRA
- $\mathbf{Segments_{CCRA}}$ - total number of touched segments in the map for CCRA

5.3. Cluj-Napoca Traffic Simulation

For measurement purposes we generated 10.000 route requests at the same time. Each route represents a car in the connected cars context. The start and end points of each request are points randomly generated inside 6 regions from Cluj-Napoca (the main Cluj-Napoca's districts).

The distance of the routes is variable and less than 10km.

The testing computer has i7-4710HQ CPU @2,50Ghz, 8GB of RAM and 30GB HDD.

Figures 1 and 2 represents the start and end points distribution in the Cluj-Napoca regions. Each ”rectangle” in the

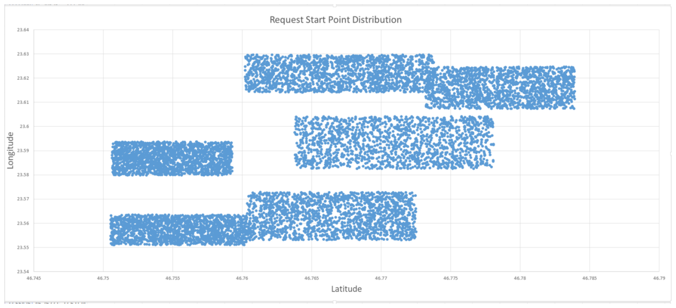

Figure 1. Route Request Start Points Distribution

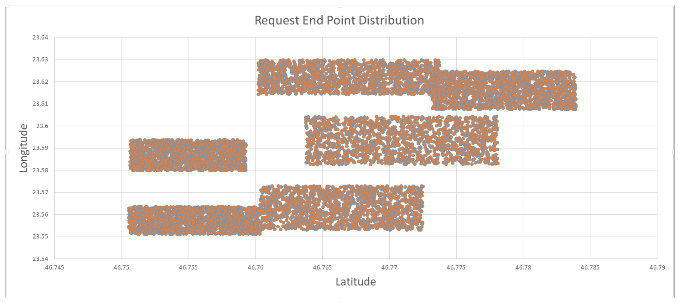

Figure 2. Route Request End Points Distribution

mentioned figures corresponds to a region in Cluj-Napoca. It can be observed that the distribution of route request start and end points is balanced Cluj-Napoca's regions.

Also, the distribution shape in figures 1 and 2 confirm that using segment trees to represent cars distribution on the map (connected cars traffic information) is the right approach.

For our test scenario we obtained $ETA_{BRA} = 23079643$ seconds and $ETA_{CCRA} = 22869461$ seconds. Comparing these two values we conclude that traffic congestion avoidance strategy in CCRA improves total spent time in traffic for 10.000 cars with more than 2.4 days meaning about 21 seconds per car on average.

The graphs in figures 3 and 4 shows that both BRA's and CCRA's routes corresponding ETAs are increasing as the number of cars in the traffic is increasing.

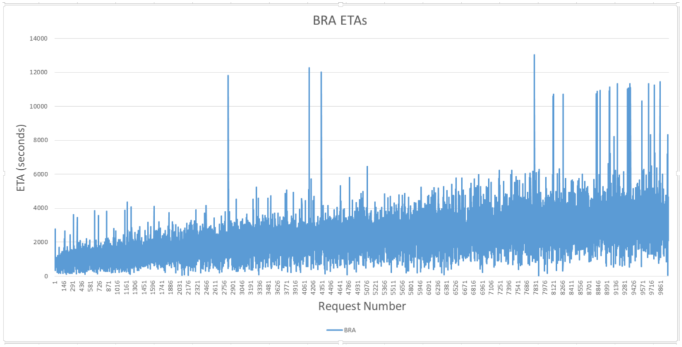

Figure 3. BRA Routes ETAs

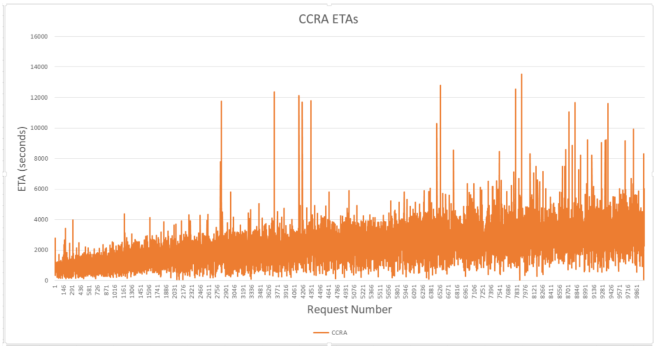

Figure 4. CCRA Routes ETAs

The obtained $Speed_{BRA} = 3.2217305m/s$ and $Speed_{CCRA} = 3.3176355$ confirms the ETA related results by showing that CCRA has a better average speed comparing with BRA.

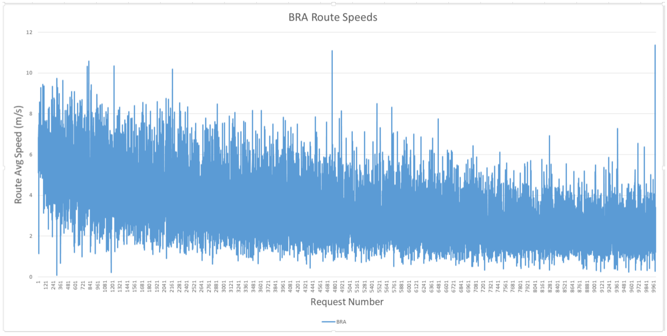

Figure 5. BRA Average Speed on Routes

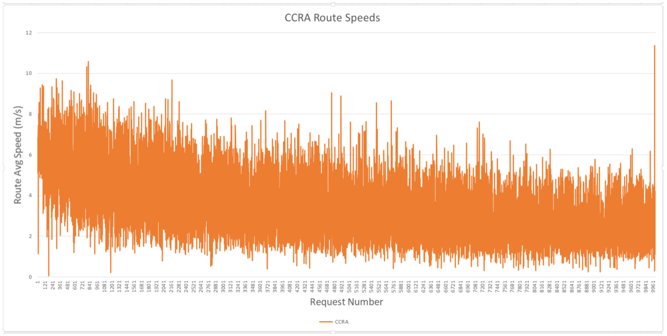

Figure 6. CCRA Average Speed on Routes

Figures 5 and 6 are mirroring figures 3 and 4 by showing that average speed on a route decreases as number of cars in traffic increases.

The number of touched segments in BRA ($Segments_{BRA} = 2777$) is less than the number of touched segments in CCRA ($Segments_{CCRA} = 2852$) proving that traffic congestion can be avoided through alternative routes that are balancing the map segments usage.

6. Related Work

The goal of creating efficient navigation systems was tackle by many, that made use of different approaches, technologies and data for their work. This resulted in different strategies that combines routing algorithms with traffic information based on connected cars data. One approach that tries to improve traffic flow through connected cars data information is found in [14]. Moreover, as shown in [15], by having a considerable amount of traffic data, there are specific patterns that can be observed and used to predict traffic congestion. However in case of not having any traffic data from traffic providers, there are also some methods in which you can generate and mimic the movement of cars inside a defined area (such an approach is described in [16]).

The common and key scope of our approach and the work in [14] and [15] is to predict and reduce the traffic congestion in different scenarios and to create different reports that would underline which are the problematic zones.

On the other hand, one important difference between our approach and the solution in [14] is the fact that we use real map data instead of synthetic map data. The experiments in [14] were made using 25.000 simulated vehicles that are positioned on a map that simulates Tokyo city in Romania. The approach in [15] uses real map data and a set of 12.000 real cars to find traffic patterns in Beijing. Our strategy uses 10.000 simulated cars that corresponds to route requests on a real map representing Cluj-Napoca city.

The approach of using Greenshieldss velocity density relationship in [14] was very effective in case of equal blocks sorted from end to start point, but this cannot be applied in our case because we use real map data. They also analyze shortest distance routes that are not in the scope of our work.

The main aspect that values our proposed strategy comparing with work in [14] and [15] is the fact that we focus to efficiently predict and avoid traffic congestion in a connected cars context, by using segment trees data structure for traffic information storage and query.

7. Conclusions and Future Work

We proposed a strategy that tries to avoid traffic congestion by using segment trees data structure to store and query traffic information in the context of connected cars. To the best of our knowledge, the usage of segment trees to store and query connected cars traffic information is a novelty in the domain.

Our solution enhances open source OSMAnd navigation software by adding simulated traffic support for their route generation algorithm. We simulated traffic information corresponding to a connected cars context and afterwards we used the traffic information in routing algorithm.

We described and implemented the Basic Routing Algorithm (BRA) that generates routes using real map data geometry and predicted traffic information. As mentioned above, segment trees data structure was used to keep and provide traffic information details in an efficient way.

As an enhancement of BRA we described and implemented Connected Cars Routing Algorithm (CCRA) that, besides using real map data geometry and predicted traffic information during route calculation, predicts traffic congestion and provides alternative routes in order to avoid traffic congestion scenarios as much as possible.

For our testing scenario we simulates 10.000 route requests representing 10.000 cars that start to navigate at the same time in 6 regions of Cluj-Napoca city. The results proved that avoiding traffic congestion can improve driving time (ETA). In our cars the improvement was of 2.4 days in total driving time for 10.000 cars that navigates on routes with variable distance and less than 10km.

The first thing we plan to do in future is to better calibrate the configuration parameters of CCRA in order to improve ETA. Another research approach that can improve ETA is to use machine learning for parameters configuration.

In terms of testing scenarios the future plan is to vary the number of simulated cars (from 1000 to 30.000) and to use different cities as driving areas. In each testing scenario we also can vary the length of the requested routes (short - smaller than 1km, medium - between 1km to 5km, long - between 5km to 20km). Last, but not least, we would like to simulate cars by introducing delays between route requests.

References

[1] F. Martinez, C.-K. Toh, J.-C. Cano, C. Calafate, and P. Manzoni, “Emergency services in future intelligent transportation systems based on vehicular communication networks,” vol. 2, pp. 6–20, 06 2010.

[2] L. Figueiredo, I. Jesus, J. A. T. Machado, J. R. Ferreira, and J. L. M. de Carvalho, “Towards the development of intelligent transportation systems,” in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, 2001, pp. 1206–1211.

[3] P. Papadimitratos, A. D. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, “Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation,” *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84–95, November 2009.

[4] J. Miller, “Vehicle-to-vehicle-to-infrastructure (v2v2i) intelligent transportation system architecture,” *2008 IEEE Intelligent Vehicles Symposium*, pp. 715–720, 2008.

[5] P. Sanders and D. Schultes, “Engineering fast route planning algorithms,” in *Experimental Algorithms*, C. Demetrescu, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 23–36.

[6] R. Gutman, “Reach-based routing: A new approach to shortest path algorithms optimized for road networks.” 01 2004, pp. 100–111.

[7] A. V. Goldberg, H. Kaplan, and R. F. Werneck, “Reach for A*: Efficient point-to-point shortest path algorithms,” in *IN WORKSHOP ON ALGORITHM ENGINEERING EXPERIMENTS*, 2006, pp. 129–143.

[8] S. Knopp, P. Sanders, D. Schultes, F. Schulz, and D. Wagner, “Computing many-to-many shortest paths using highway hierarchies,” 01 2007.

[9] “Routing server - waze,” accessed: 2018-06-14. [Online]. Available: https://wiki.waze.com/wiki/Routing_server

[10] V. Shcherb, “Osmand,” accessed: 2018-06-14. [Online]. Available: https://osmand.net/

[11] W. Zeng and R. L. Church, “Finding shortest paths on real road networks: the case for a*,” *International Journal of Geographical Information Science*, vol. 23, no. 4, pp. 531–543, 2009. [Online]. Available: https://doi.org/10.1080/13658810801949850

[12] D. Delling and G. Nannicini, “Bidirectional core-based routing in dynamic time-dependent road networks,” in *Algorithms and Computation*, S.-H. Hong, H. Nagamochi, and T. Fukunaga, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 812–823.

[13] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008.

[14] T. Yamashita, K. Izumi, K. Kurumatani, and H. Nakashima, “Smooth traffic flow with a cooperative car navigation system,” in *Proceedings of the International Conference on Autonomous Agents*, 01 2005, pp. 478–485.

[15] J. Wang, Y. Mao, J. Li, Z. Xiong, and W.-X. Wang, “Predictability of road traffic and congestion in urban areas,” *PLOS ONE*, vol. 10, no. 4, pp. 1–12, 04 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0121825

[16] J. Capela, P. Henriques Abreu, D. Castro Silva, G. Fernandes, P. Machado, and A. Leito, “Preparing data for urban traffic simulation using sumo,” 05 2013.